Name: Feifan Liao
ID:fl2656 Work with：Xingtong Huang, Zhimei Chen, Jiahang Liu

Problem 1b: Understand BERT Inputs

Answer:
According to the original paper and code results, each of those three inputs represent:

1.input_ids

is the numerical IDs ("token embeddings" in the original paper) of the text based on BERT's vocabulary (including special tokens like [CLS] 101, [SEP] 102 and [PAD] 0). For example, by reading the code cells, we can know that "Hello" = 7592, "!" = 999 and so on. And the original first sentence is one token shorter than the second one (5 < 6, including [CLS] and [SEP]), so the code add a padding "0" to its input_ids.

2.token_type_ids

is the segment IDs ("segment embeddings" in the original paper) to help model to differentiate whether a token belongs to the first sentence (0) or the second sentence (1) in text-pairs (if the task only involve single text, then they are all "0" in token_type_ids). For example, by reading the code cells, we can know that this is a single-text case, "Hello world!" and "How are you?" are both have the token_type_ids of "0"s.

3.attention_mask

is a tensor that indicates which tokens should be attended to during BERT’s self-attention computation. To be specific, the valid part of the tokens will have the attention_mask of "1", the rest of the tokens will be "0" (sould be ignored, like [PAD]). For example, by reading the code cells, all of the tokens are valid expect the [PAD] in the first sentence, when the model process the tokens, it will ignore the [PAD] (or "0").

Problem 1c: Understand BERT Hyperparameters

According to the original paper and the official BERT Github repository, the BERT_tiny model was trained by pre-training, fine-tuning, and knowledge distillation (the authors call this method "Pre-trained Distillation (PD)"). From the doucumentation in the official repository, the best fine-tuning hyperparameters were selected from:
batch sizes = 8, 16, 32, 64, 128
learning rates = 3e-4, 1e-4, 5e-5, 3e-5
and each combination of hyperparameters were trained for 4 epochs.

Problem 3a: Train Models

	Validation Accuracy	**Learning Rate**	**Batch Size**
Without BitFit	0.890	3e-4	64
With BitFit	0.632	3e-4	8

Problem 3b: Test Models and Report Results

	# Trainable Parameters	Test Accuracy
Without BitFit	4386178	0.878
With BitFit	3074	0.627

1.According to the table above, after we apple bitFit method, the trainable parameters will be reduced by about 99.93%, and test accuracy drops by about 24% in IMDb. Therefore, for BERT_tiny in IMDb task, full fine-tuning performs better than bitFit.
2.The results reported by Zaken et al. (2020) shows that bitfit works better in smaller datasets (performs comparably to full fine-tuning), but will noticeable drop accuracy on larger datasets. In Zaken et al's paper, there are large datasets like MNLI(393k) and QQP(364k), and small datasets like RTE(2.5k) and CoLA(8.5k), compared to these datasets, our IMDb (about 50k) could be considered as medium-to-large scale dataset, so the accuracy drop with bitfit align with paper's finding.
3.In general, there is not double that bitFit's powerful ability to reduce computational burden is suitable for resource-constrained environments and small-scale datasets, but the accuracy drop compared to full fine-tuning makes it not a quite ideal method if we have high-performance devices and need to deal with larger datasets like IMDb.